IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Mitchell Alsup, et al.	§	Examiner: Fennema, Robert E.
	§	Group Art Unit: 2183
Serial No. 10/726,902	§	
	§	Atty. Dkt. No.: 5500-88700
Filed: December 3, 2003	§	
For: Transitioning from Instruction Cache to Trace Cache on Label Boundaries	§	

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed June 30, 2008, Appellants present this Appeal Brief. Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

I. REAL PARTY IN INTEREST

As evidenced by the assignment recorded at Reel/Frame 014767/0579, the subject application is owned by Advanced Micro Devices, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at One AMD Place, P.O. Box 3453, Sunnyvale, CA 94088.

II. <u>RELATED APPEALS AND INTERFERENCES</u>

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. <u>STATUS OF CLAIMS</u>

Claims 1-35 are pending in the application and stand finally rejected. The rejection of claims 1-35 is being appealed. A copy of claims 1-35 is included in the Claims Appendix herein below.

IV. STATUS OF AMENDMENTS

No amendments have been submitted subsequent to the last rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a microprocessor that includes an instruction cache, a branch prediction unit, a trace cache, and a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache. (*See, e.g.,* FIG. 1, microprocessor 100, instruction cache 106, branch prediction unit 132, trace cache 160, and prefetch unit 108; and p. 7 lines 3-5 and 18-27.)

The instruction cache is configured to store instructions. *(See, e.g.*, p. 7, lines 18-18-22; and p. 8, lines 15-16.)

The trace cache is configured to store a plurality of traces of instructions. (*See, e.g.,* FIG. 2, trace cache entry 162; p. 10, lines 18-29; and p. 13, lines 6-9.)

The prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address. (*See, e.g.,* p. 15, lines 5-15; FIG. 3, block 301, and the feedback loop from the negative exit of decision block 303 to block 301; and p. 15, line 27 – p. 16, line 3.)

The prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address. (*See, e.g.,* FIG. 3, the positive exit of decision block 303, and block 305; and p. 16, lines 5-9.)

The prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address. (*See, e.g.,* FIG. 3, the negative exit of decision block 303 to block 301; p. 15, line 27 – p. 16, line 3; and p. 18, lines 9-23.)

In response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the traces from the trace cache. (*See, e.g.,* FIG. 3, the positive exit of decision block 305, and block 307; and p. 16, lines 9-15.)

Independent claim 14 is directed to a computer system that includes a system memory and a microprocessor coupled to the system memory. (*See, e.g.,* FIG. 1, microprocessor 100 coupled to system memory 200; and p. 7, lines 18-28.)

The microprocessor includes an instruction cache, a branch prediction unit, a trace cache, and a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache. (*See, e.g.,* FIG. 1, microprocessor 100, instruction cache 106, branch prediction unit 132, trace cache 160, and prefetch unit 108; and p. 7 lines 3-5 and 18-27.)

The instruction cache is configured to store instructions. *(See, e.g.*, p. 7, lines 18-18-22; and p. 8, lines 15-16.)

The trace cache is configured to store a plurality of traces of instructions. (*See, e.g.,* FIG. 2, trace cache entry 162; p. 10, lines 18-29; and p. 13, lines 6-9.)

The prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address. (*See, e.g.,* p. 15, lines 5-15; FIG. 3, block 301, and the feedback loop from the negative exit of decision block 303 to block 301; and p. 15, line 27 – p. 16, line 3.)

The prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address. (*See, e.g.,* FIG. 3, the positive exit of decision block 303, and block 305; and p. 16, lines 5-9.)

The prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address. (*See, e.g.,* FIG. 3, the negative exit of decision block 303 to block 301; p. 15, line 27 – p. 16, line 3; and p. 18, lines 9-23.)

In response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the traces from the trace cache. (*See, e.g.,* FIG. 3, the positive exit of decision block 305, and block 307; and p. 16, lines 9-15.)

Independent claim 27 is directed to a method that includes receiving a retired instruction. (*See, e.g.,* p. 12, line 27- p. 13, line 3; p. 16, lines 26-29; and p. 17, lines 11-13.)

The method also includes determining if a previous trace under construction duplicates a trace in a trace cache and if the received instruction corresponds to a branch label. (*See, e.g.,* FIG. 4, decision blocks 353 and 357; and p. 19, lines 18-27.)

The method further includes, in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label, beginning construction of a new trace. (*See, e.g.,* FIG. 4, the positive exits from decision blocks 353 and 357, and block 359; and p. 19, lines 29-31.)

Independent claim 32 is directed to method that includes fetching instructions from an instruction cache. (*See, e.g.,* p. 15, lines 5-15; FIG. 3, block 301 and p. 15, line 27-28.)

The method includes continuing to fetch instructions from the instruction cache without searching a trace cache until a branch target address is generated. (*See, e.g.,* p. 15, lines 5-15; FIG. 3, block 301, and the feedback loop from the negative exit of decision block 303 to block 301; FIG. 3, the negative exit of decision block 303 to block 301; p. 15, line 27 – p. 16, line 3; and p. 18, lines 9-23.)

The method further includes, in response to a branch target address being generated, searching a trace cache for an entry corresponding to the branch target address. (*See, e.g.,* FIG. 3, the positive exit of decision block 303, and block 305; and p. 16, lines 5-9.)

Independent claim 35 is directed to a microprocessor that includes means for receiving a retired operation. (*See, e.g.,* FIG. 1, trace generator 170, which is configured to receive basic blocks of retired operations from retire queue 102; p. 12, line 27 - p. 13, line 3; p. 16, lines 26-29; and p. 17, lines 11-13.)

The microprocessor also includes means for determining if a previous trace under construction duplicates a trace in a trace cache and if the received operation is a first operation at a branch label. (*See, e.g.,* FIG. 1 trace generator 170; FIG. 4, decision blocks 353 and 357; and p. 19, lines 4-16 and 18-27.)

The microprocessor further includes means for starting a new trace in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received operation is a first operation at a branch label. (*See, e.g.,* FIG. 1, trace generator 170; FIG. 4, the positive exits from decision blocks 353 and 357, and block 359; and p. 19, lines 29-31.)

The summary above describes various examples and embodiments of the claimed subject matter; however, the claims are not necessarily limited to any of these examples and embodiments. The claims should be interpreted based on the wording of the respective claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 2, 11-15, 24-26, and 32-34 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, et al. (hereinafter "Rotenberg") in view of Xia.

2. Claims 3 and 16 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg and Xia in view of Patterson, et al. (hereinafter "Patterson").

3. Claims 4, 10, 17 and 23 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg and Xia in view of Braught.

4. Claims 5-8 and 18-21 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, Xia, Braught and further in view of Lange, et al. (U.S. Patent 3,896,419) (hereinafter "Lange").

5. Claims 9 and 22 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, Xia and Braught in view of Akkary, et al. (U.S. Patent 6,247,121) (hereinafter "Akkary").

6. Claims 27-31 and 35 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, Xia, Braught and Lange in view of Akkary.

VII. ARGUMENT

First ground of rejection:

The Examiner rejected claims 1, 2, 11-15, 24-26, and 32-34 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, et al. (hereinafter "Rotenberg") in view of Xia. Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 2, 12, 14, 15, 25, and 32-34:

1. The cited art clearly fails to teach or suggest wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address, wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address, wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address, as recited in claim 1.

The Examiner submits that these limitations are inherent in the system of Rotenberg. However, in the system of Rotenberg, "The trace cache is accessed in parallel with the instruction cache and BTB using the current fetch address..." (emphasis added.) **It is clear in this and other passages of Rotenberg, that in Rotenberg every fetch address is compared to the entries in the trace cache, regardless of the operation of the branch prediction unit.** Rotenberg teaches searching for a hit in the trace cache on every cycle, and then fetching from the instruction cache if there is not a hit in the trace cache.

By contrast, Appellants' claim 1 recites that the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting a predicted target address, and that the trace cache is not checked for a match until the branch prediction unit outputs the predicted target address. In other words, the trace cache is not checked for a match on every cycle, but only on cycles for which the branch prediction unit outputs a predicted target address. The Examiner previously submitted, "It is inherent that you cannot check for a value if you do not know what the value is. The trace cache as disclosed by Rotenberg cannot search for the predicted target address in the cache if it does not know what the predicted target address is, and in fact, no cache or any other type of hardware can find something when it doesn't know what it is looking for, thus, the value must have been output from the branch prediction unit prior to the checking." Appellants assert that the Examiner's interpretation of the operation of Rotenberg is **demonstrably incorrect.** There is no need in Rotenberg to have a *predicted* target address in order to check for a match. The predicted target address is not what is being compared in Rotenberg. Rotenberg explicitly checks for a match on every cycle, regardless of the operation of the branch prediction unit and whether it outputs a predicted target address.

In the Final Action mailed July 18, 2007, the Examiner submitted, "When looking at the entire claim in context, it can be seen that the claim is referring to searching the cache for a match for the predicted target address, and that it cannot search the cache for the predicted target address until the predicted target address is generated" (emphasis Examiner's). **The Examiner has misread the claim.** The referenced limitation of the claim does not recite "not check the trace cache for the match" (i.e., referring to the match for the predicted target address of the previous limitation), but instead recites "not check the trace cache for a match" (i.e., any match between the current address and the trace cache). **The plain language of the claim does not support the Examiner's interpretation. As explicitly recited in claim 1,** multiple instructions are fetched from the instruction cache, **without checking for a match (hit) in the trace cache**, until the branch prediction unit outputs a predicted branch address (i.e., until a branch instruction is encountered for which the branch prediction unit outputs a predicted target address). Only then (**in response to this output**) is the trace cache checked for a match. This functionality is clearly not suggested by any evidence of record.

Appellants again assert that one of ordinary skill in the art having benefit of Appellants' disclosure could not agree with the Examiner's interpretation of the referenced limitation. The Examiner's repeated assertions that it is impossible to search for something that has not yet been generated simply shown a misunderstanding on the Examiner's part of what is described in the cited art. The Examiner's assertions are irrelevant to Appellants' argument and to what is actually recited in the claim. The Examiner's interpretation is also completely inconsistent with the specification (see, e.g., FIG. 3 and pages 15-18). Also, in remarks in the Final Action regarding claim 32, the Examiner admitted that Rotenberg teaches that the trace cache is searched <u>on every instruction</u>, which is clearly not <u>in response to the branch prediction unit outputting a predicted target address</u>, as required by claim 1.

2. **The cited art clearly fails to teach or suggest *wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address*, as recited in claim 1.**

In the Office action mailed March 31, 2008, the Examiner admits that Rotenberg fails to teach this limitation and relies on Xia to teach it, citing Section 5.5, first scheme. Specifically, the Examiner submits "Xia teaches that an advantage of starting a trace only at predictable branch instructions… is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme)."

Appellants again note that the Examiner has admitted that Rotenberg teaches that the trace cache is <u>searched on every instruction</u>. **Appellants also assert that Xia's trace table and instruction cache are also <u>checked in parallel on every instruction</u>.** There is nothing in Xia that describes how, or more importantly <u>when</u>, the trace cache is <u>checked for the start of a trace line</u>, or when the system begins a search of the trace cache for any reason. The Examiner's statement made in remarks regarding claim 32 in the Final Action of July 18, 2007, "The advantages that Examiner indicated… are obvious advantages one of ordinary skill in the art would recognize, searching a cache takes time and energy, and if there is no possible way that searching a cache can benefit the user, there is absolutely no reason to do so" **contradict the teachings of his own references, which <u>search the trace cache for a hit</u> (although not necessarily a trace <u>start</u>) <u>on every cycle</u>. This is in direct conflict with the limitations of claim 1.**

3. The Examiner has failed to provide a proper reason to combine the references.

In the Office action mailed March 31, 2008, the Examiner submits, "Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on braches. In addition, because traces would only thus be started on branches, it wouldn't make sense to look for a hit in the trace cache on non-branches, since it couldn't possibly result in a hit, thus one of ordinary skill in the art would not search a trace cache when it is not required, and save power by not having to constantly access the cache with no hope of a hit."

In the Response to Arguments section of the Office Action mailed March 31, 2008, the Examiner restates his argument this way, "Xia teaches a reason as to why one would only want to start a trace on a branch (smaller cache size), and Examiner has said that one of ordinary skill in the art would be able to take that knowledge, and when incorporating it into Rotenberg, realize that if traces only start on branches, that it is impossible to get a trace cache hit when the instruction is not a branch, thus it would be a waste of power to even try (and may also cause critical path problems, issues that those of ordinary skill in the art would clearly be able to recognize, and thus would implement the invention as claimed, because it is an obvious modification based on the information provided by the references."

As discussed above, the Examiner's stated reason for combining the references in teaching the above-referenced limitation, i.e. his assertion that such a

technique would contribute to power saving or critical path reduction, is completely unsupported in the cited art. The only advantage noted for starting traces on branches is to save memory space. In addition, such a change does not require or even suggest a change to the method for searching the trace cache, as the Examiner incorrectly contends. Furthermore, there is no evidence of record that changing the search method, as suggested by the Examiner (but not by either of the cited references), would necessarily contribute to power savings or critical path reduction in a system that includes the combination suggested by the Examiner, as this would be dependent on the specific circuitry used to implement such a combination. The Examiner's assertion, therefore, is nothing but pure speculation.

4. Even if the references were combined, their combination fails to teach or suggest the claimed invention.

In the Response to Arguments section of the Office Action mailed March 31, 2008, the Examiner also states, "Examiner is using Xia to teach the advantages of starting traces on branches, and is not attempting to bodily incorporate the two references... Examiner reminds Applicant that the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art." **Appellants again assert that the Examiner's repeated contention that it would be obvious not to search the trace cache for a non-branch instruction in a system that incorporates the trace-starting feature of Xia is completely unsupported by the cited art, as his own references (including Xia) check for a match on every instruction cycle.** It is unclear how the referenced feature of Xia can possibly teach or suggest the above-referenced limitation of claim 1 in the combined system if it does not teach or suggest this limitation in its own system.

As discussed at length above, Appellants assert that even if the referenced feature of Xia (i.e., starting traces only on branches) were incorporated into the system of Rotenberg, this would not teach or suggest any change to the **search** method of the combined system, much less a change that is not taught or suggested by either reference. **At most, it would result in a system in which traces begin on branches, as described in Xia, but in which the trace cache of Rotenberg (and/or the trace table and instruction cache of Xia) are still checked in parallel on every instruction.**

For at least the reasons above, the rejection of claim 1 is unsupported by the cited art and removal thereof is respectfully requested.

Independent claims 14 and 32 include limitations similar to claim 1, and so the arguments presented above apply with equal force to this claim, as well.

Claims 11 and 24:

1. **The cited art clearly fails to disclose *wherein each of the plurality of traces comprises partially-decoded instructions*, as recited in claim 11.**

The Examiner previously submitted that it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as shown in section 2.2 of Rotenberg would not be available. Appellants asserted, however, that the control information described in section 2.2 is control information added to a trace cache entry when the trace cache entry is generated and stored along with the instructions themselves in the trace cache. For example, Rotenberg describes that the control information is "similar to the tag array of standard caches but contains additional state information." The control information described in section 2.2 comprises control information generated by the branch predicator and other logic of the trace cache, not information decoded from the instruction itself. **Therefore, the Examiner's assertion that traces inherently comprise partially-decoded instructions is clearly incorrect.** Nothing in Rotenberg describes that the instructions in a trace entry must be partially decoded in order to generate the control information, nor would generating the control information described in section 2.2 require such decoding.

In the Response to Arguments section of the Office Action of March 31, 2008, the Examiner also submits, "However, these traces clearly hold data that cannot be obtained without some form of decoding, for example, the traces hold the start and next addresses of the trace, information which is impossible to obtain without decoding, since an instruction is just a random collection of bits until it is decoded and put into a context. The branch flags and branch mask bits also require decoding, since simply identifying that there is even one branch in the trace requires that that branch instruction was decoded to identify it as a branch." Appellants' argument is not that decoding is not ever performed on instructions, but that the instruction traces themselves do not comprise partially decoded instructions. Section 2.2 of Rotenberg states, "The trace cache is made up of instruction traces, control information, and line-fill buffer logic," clearly making a distinction between instruction traces and the control information described above (e.g., the start and next addresses, branch flags, etc.) **This control information is not part of the instruction traces themselves.** The instruction traces store instructions that are output to a mux (shown in FIG. 4), and are not described as included decoded information. In fact, they are depicted as passing to an instruction latch as un-decoded instructions. This is clearly illustrated in FIG. 4 by the fact that the stored instructions pass from the instruction latch to the decoder.

For at least the reasons above, the rejection of claim 11 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 24 includes limitations similar to claim 11, and so the arguments presented above apply with equal force to this claim, as well.

<u>**Claims 13 and 26:**</u>

1. **The cited art clearly fails to disclose *wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace,* as recited in claim 13.**

The Examiner previously submitted that Rotenberg teaches this limitation in Section 2.2, and that the "trace target address" and "trace fall-through address" are labels that describe where control will flow based on each branch operation, based on a prediction. However, these addresses are actually described as corresponding to the next fetch addresses if the <u>last branch in the trace</u> is predicted taken or not taken, respectively. There is nothing in Rotenberg that describes labels for an instruction to which control will pass for <u>each branch operation comprised in the trace</u>, as recited in claim 13, only for the <u>last branch in the trace</u>.

In the Office Action of March 31, 2008, the Examiner submits that Section 2.2 of Rotenberg teaches, "the branch flags exist for every branch in the trace, and indicate which instruction control will pass to (the taken or not taken))." This passage describes that each trace includes a branch flag for each branch within a trace indicating which path was taken by the actual trace (taken or not taken) for that branch. It does not, however, describe that a label for an instruction to which control will pass is included for each instruction. **Instead, as previously noted, the trace includes a trace target address and a trace fall-through address <u>only for the last branch</u> in the trace.**

For at least the reasons above, the rejection of claim 13 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 26 includes limitations similar to claim 13, and so the arguments presented above apply with equal force to this claim, as well.

Second ground of rejection:

The Examiner rejected claims 3 and 16 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg and Xia in view of Patterson, et al. (hereinafter "Patterson"). Appellants traverse this rejection for at least the following reasons.

1. The cited art clearly fails to teach or suggest *wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction,* as recited in claim 3.

The Examiner admits that Rotenberg fails to disclose this limitation and relies on Patterson to teach it. The Examiner submits that Patterson teaches, "in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to immediately output the correct path on a misprediction without having to fetch it (Pages 276-277)." This passage actually describes reducing the penalty for a misprediction by <u>fetching instructions from both the predicted and unpredicted directions</u>, e.g., by using a dual-ported cache, an interleaved cache, or by fetching from one path and then the other (i.e., fetching from both paths <u>before determining whether a branch has been mispredicted</u>). It does not describe <u>outputting a predicted target address</u> (to be checked against entries in a trace cache, as in Appellants' claims) <u>in response to detection of a branch misprediction</u>.

On page 11 and in the Response to Arguments section of the Office Action of March 31, 2008, the Examiner submits that one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increase performance "by allowing the other path to be immediately output when the misprediction is noted" and "in the combination proposed, the branch predictor holds both the taken and not taken paths, therefore, on a misprediction, the other path is available to be output immediately, thus teaching the limitation." Appellants note that claim 3 does not recite a branch predictor holding both the taken and not taken branch paths so that "the other path is available to be output immediately", but explicitly recites outputting the predicted target address **in response to** detection of a branch misprediction. There is nothing in the Examiner's citation that describes a mechanism for outputting the predicted target address in response to detection of a branch misprediction.

For at least the reasons above, the rejection of claim 3 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 16 includes limitations similar to claim 3, and so the arguments presented above apply with equal force to this claim, as well.

Third ground of rejection:

The Examiner rejected claims 4, 10, 17 and 23 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg and Xia in view of Braught. Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 4 and 17:

For at least the reasons presented herein regarding the claims from which they depend, the rejection of claims 4 and 17 is unsupported by the cited art, and removal thereof is respectfully requested.

Claims 10 and 23:

1. The cited art clearly fails to teach or suggest *wherein the trace generator is configured to generate traces in response to instructions being decoded,* as recited in claim 10.

The Examiner submits that Rotenberg teaches this limitation in Section 2.2, "the trace can not be completed (written into the cache) until the trace target addresses are calculated, which in turn requires the instructions to be decoded." Appellants assert that this does not teach the above-referenced limitation of Appellants' claim, in which traces are generated (not just completed) in response to instructions being decoded. There is nothing in Rotenberg that describes trace generation being performed in response to instructions being decoded.

In the Response to Arguments section of the Office Action of March 31, 2008, the Examiner references remarks made regarding claim 11, "which shows that the trace requires decoded instructions in order to be created." However, as discussed above, the cited art does not teach that traces comprise partially decoded instructions. In addition, even if instructions included in traces are decoded at some point, there is nothing in the Examiner's citations that teaches or suggest **generating traces in response to this decoding**, as required by claim 10.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claims 10. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

For at least the reasons above, the rejection of claim 10 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 23 includes limitations similar to claim 10, and so the arguments presented above apply with equal force to this claim, as well.

Fourth ground of rejection:

The Examiner rejected claims 5-8 and 18-21 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, Xia, Braught and further in view of Lange, et al. (U.S. Patent 3,896,419) (hereinafter "Lange"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 5 and 18:

1. The cited art clearly fails to teach or suggest *wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing,* as recited in claim 5.

The Examiner admits that Rotenberg, Xia, and Braught fail to teach this limitation and relies on Lange to teach it. As discussed below regarding claims 27 and 35, Lange's description of the operation of a data cache during fetching from main memory has absolutely nothing to do with the specific limitations of claim 5. Appellants assert that identifying a duplicate copy of a value in a cache when merely fetching a value from memory is clearly not analogous to identifying a trace entry corresponding to a trace entry under construction (e.g., in response to decoding and/or execution of previously fetched instructions), as in Appellants' claimed invention. Nothing in Lange teaches or suggests a mechanism for the identification of such trace entries.

Checking Rotenberg's trace cache for a duplicate trace before collecting a new trace is also contradictory to the Examiner's remarks regarding the advantages of including duplicate traces. **The Examiner first argues that Rotenberg teaches duplicate traces may exist, and then argues that the combined references teach that the system of Rotenberg should not allow construction of duplicate traces. These arguments cannot coexist if the references are to teach the limitations of claim 5. If**

no duplicate traces can be constructed, there would be no reason to check the trace cache for a duplicate copy of a trace that the trace generator is constructing.

In the Response to Arguments section of the Office Action of March 31, 2008, the Examiner states that Applicant has argued that the references do not teach the limitations of claim 5 and that Applicant has argued that Rotenberg does not allow for duplicate traces, thus cannot claim them. The Examiner has mischaracterized Appellants' argument above. Appellants' argument above is that the Examiner's remarks regarding the teaching of claim 5 <u>contradict each other</u>, as discussed in more detail below, and therefore do not establish a *prima facie* obviousness of the claimed invention.

2. The Examiner has not provided a valid reason to combine the references.

The Examiner submits that one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg in order to solve either or both of the problems of getting a miss while servicing a miss (in a system that <u>allows duplicate traces</u>), and preventing eviction of a useful trace by a duplicate trace (in a system that <u>does not allow duplicate traces</u>). These are clearly contradictory goals with contradictory solutions. It is not clear how the first goal (i.e. handling getting a cache miss while servicing a miss) could solved by checking for duplicate traces and discarding duplicates in a system that allows duplicate tracing, as suggested by the Examiner. **First, handling a cache miss while servicing another cache miss has nothing to do with the limitations of claim 5, which is directed to <u>trace generation</u>.** Furthermore, if the solution to this problem involved detecting duplicate traces during construction and then discarding them, the system would, in fact, <u>not allow duplicate traces</u>, in contrast to the Examiner's suggestion. Thus, it would not be consistent with the "judicial trace selection" example described below in remarks regarding claim 27. In addition, as discussed above, the referenced feature of Lange does not describe checking for <u>a duplicate copy of a trace while constructing a trace</u>, as it has nothing to do with the operation of a trace cache at all. Therefore, it cannot be used to solve the second problem cited by the Examiner (preventing eviction of a useful trace by a duplicate trace).

The Examiner's reasons for combining the references are clearly not supported by the cited art, and the references, when combined, do not teach the limitation recited in claim 5.

For at least the reasons above, the rejection of claim 5 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 18 includes limitations similar to claim 5, and so the arguments presented above apply with equal force to this claim, as well.

Claims 6 and 19:

1. The cited art clearly fails to teach or suggest *wherein in response to the trace generator identifying a duplicate copy of the trace, the trace generator is configured to discard the trace under construction,* as recited in claim 6.

The Examiner previously cited Lange as teaching this limitation in column 5, lines 5-10. Appellants have argued that, as discussed herein, Lange is not directed to trace generation at all. Lange's description of the operation of a data cache during fetching from main memory has absolutely nothing to do with the specific limitations of Appellants' claims. For example, there is nothing in the combination of cited references that teaches or suggests identifying a duplicate copy of a trace and in response to such identification, discarding a trace under construction.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 6. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

Appellants note that the Examiner did not provide any additional remarks regarding claim 6 in the Office Action of March 31, 2008.

For at least the reasons above, the rejection of claim 6 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 19 includes limitations similar to claim 6, and so the arguments presented above apply with equal force to this claim, as well.

<u>**Claims 7 and 20:**</u>

1. The cited art clearly fails to teach or suggest *wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated,* as recited in claim 7.

The Examiner previously asserted that Rotenberg teaches this limitation in Section 2.2, "The trace cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again." **Appellants asserted that this clearly does not teach the specific limitations of Appellants' claim, as Rotenberg does not perform the checking <u>in response to identifying a duplicate copy of the trace</u>, as required by claim 7.** Instead, "the trace cache is checked every time there is a potential new trace" regardless of whether a duplicate copy has been identified.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 7. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

Appellants note that the Examiner did not provide any additional remarks regarding claim 7 in the Office Action of March 31, 2008.

For at least the reasons above, the rejection of claim 7 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 20 includes limitations similar to claim 7, and so the arguments presented above apply with equal force to this claim, as well.

<u>**Claims 8 and 21:**</u>

1. The cited art clearly fails to teach or suggest *wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction,* as recited in claim 8.

The Examiner previously submitted that Lange teaches this limitation in column 5, lines 5-10. **Appellants have argued that, as discussed above, Lange teaches nothing about the generation of <u>trace entries</u>, much less the <u>specific limitations of claim 8</u>.** For example, there is nothing in the combination of references that teaches or suggests <u>discarding a trace under construction in response to identifying a trace entry</u> corresponding to the <u>next trace to be generated</u> (i.e., a trace other than the current trace entry).

In the Response to Arguments section of the Office Action of March 31, 2008, the Examiner submits, "Lange teaches an analogous situation to the claim, where a duplicate copy of a value in a cache is discarded when it is found to be a duplicate, which is what the claim is claiming." Appellants again assert that identifying "a duplicate copy of a value in a cache" when merely fetching a value from memory is clearly not analogous to identifying a trace entry corresponding to <u>a trace entry under construction</u> (e.g., in response to decoding and/or execution of previously fetched instructions) much less to identifying the next trace to be generated, during construction of a current trace (from executed instructions). Nothing in Lange teaches or suggests a mechanism for the identification of such trace entries.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 8. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

In the Response to Arguments section of the Office Action of March 31, 2008, the Examiner submits, "Additionally, Examiner has already laid out a motivation for combining Lange with Rotenberg, and Examiner refers the Applicant to the rejection to see said motivation." However, as discussed above, the referenced feature of Lange is not analogous to the limitations recited in claim 8, or in others of the dependent claims.

For at least the reasons above, the rejection of claim 8 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 21 includes limitations similar to claim 8, and so the arguments presented above apply with equal force to this claim, as well.

Fifth ground of rejection:

The Examiner rejected claims 9 and 22 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, Xia and Braught in view of Akkary, et al. (U.S. Patent 6,247,121) (hereinafter "Akkary"). Appellants traverse this rejection for at least the following reasons.

1. The cited art clearly fails to teach or suggest *wherein the trace generator is configured to generate traces in response to instructions being retired,* as recited in claim 9.

The Examiner admits that Rotenberg does not teach that instructions need to be retired before the trace can be generated and relies on Akkary to teach this limitation. The Examiner submits that Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (column 3, lines 40-44). **<u>This passage actually says just the opposite</u>**, that is, that instructions placed in the trace buffer <u>stay in the trace buffer until they are retired</u>.

In the Response to Arguments section of the Office Action of March 31, 2008, the Examiner submits, "Applicant is attempting to bodily incorporate the references, as opposed to looking at the rejection that the Examiner has laid out. Akkary teaches that retired instructions are guaranteed to be correct, and Rotenberg teaches that putting incorrect instructions into the trace cache can cause significant problems, therefore the Examiner has said it would have been obvious to one of ordinary skill in the art to wait until an instruction is retired in order to place it in the trace cache, and the manner in which Akkary uses a trace buffer is completely unrelated to the rejection, and it is improper to bodily incorporate the references as the Applicant is attempting to do to argue the rejection of the claim." **Appellants assert that the argument above is directed to the Examiner's own remarks**. The Examiner submitted, "Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly." **This is incorrect. Akkary teaches putting instructions in the trace buffers <u>before retirement</u> (i.e., before they are known to be correct).** Thus, Akkary teaches away from the limitations of Appellants' claim 9. Appellants remind the Examiner, "It is improper to combine references where the references teach away from their combination." *In re Grasselli*, 218 USPQ 769, 779 (Fed. Cir. 1983).

Appellants further assert that the Examiner appears to be relying on each reference individually. This is the hallmark of hindsight reconstruction. It is not proper to pick and chose isolated teachings from references to reconstruct an applicant's claim. Instead, the Examiner must consider the references as a whole, including parts that teach away. M.P.E.P. § 2141.02, last paragraph; *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 851 (1984). **Since Akkary explicitly teaches that instructions are put into the trace buffers before retirement, Akkary cannot be properly combined with Rotenberg to teach *wherein the trace generator is configured to generate traces in response to instructions being retired.***

For at least the reasons above, the rejection of claim 9 is unsupported by the cited art, and removal thereof is respectfully requested.

Claim 22 includes limitations similar to claim 9, and so the arguments presented above apply with equal force to this claim, as well.

Sixth ground of rejection:

The Examiner rejected claims 27-31 and 35 under 35 U.S.C. § 103(a) as being unpatentable over Rotenberg, Xia, Braught and Lange in view of Akkary. Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Independent Claims 27 and 35:

1. The cited art clearly fails to teach or suggest *receiving a retired instruction.*

The Examiner previously admitted that Rotenberg does not teach that instructions need to be retired before the trace can be generated and relies on Akkary to teach this limitation. The Examiner submitted that Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (column 3, lines 40-44). **However, as discussed above, this passage actually says just the opposite**, that is, that instructions placed in the trace buffer stay in the trace buffer until they are retired. Therefore, Akkary teaches away from the limitation above.

2. The cited references fail to teach or suggest *in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label, beginning construction of a new trace.*

The Examiner admitted that Rotenberg fails to teach this limitation or to discuss the issue of duplication. The Examiner submitted that Rotenberg discusses the disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and teaches the disadvantages of a useless trace displacing a useful trace. The Examiner then submitted, "Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss." Appellants asserted that the Examiner is contradicting himself and that he has cited nothing in Rotenberg that teaches tracing of potential duplicate traces. In Rotenberg, the fill-line buffer logic services trace cache misses (i.e., the combination of a matching target address and matching branch flags is not found in the trace cache). There is no reason to believe there would ever be a duplicate trace in the system of Rotenberg, and no reasons or opportunity to avoid such duplication. The Examiner suggested that there is a "potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments." **This is completely unsupported in Rotenberg and does nothing to overcome the deficiencies of the cited references in teaching the above-referenced limitations. Similarly, the Examiner's contention that "Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used" is also completely unsupported.** The solution discussed in this section is completely different from the specific limitations recited in claim 27. The Examiner submitted that this section provides motivation to handle cases involving duplicate traces, and that Lange provides such a method, by simply discarding the duplicate value. **Again, the Examiner's remarks do not address the limitations recited in claim 27, which are directed to specific conditions to be met before beginning construction of a new trace, and which the cited references do not teach.**

3. The Examiner has failed to provide a proper reason to combine the references.

The Examiner has argued that Akkary suggests the limitation of using a retired instruction, and he includes remarks regarding "correctness" of a trace, a goal of Rotenberg's trace cache to exploit code reuse, and that branches tend to be biased in one direction. However, Rotenberg explicitly states that the trace line is filled <u>as instructions are fetched from the instruction cache</u>, **which is clearly incompatible with filling the trace line with retired instructions**. The fact that the system of Rotenberg <u>could</u> solve this problem in a manner different from the only example described does not suggest <u>the specific solution recited in the claims</u>. **Appellants also asserted that combining the references does not teach the claimed invention,** as <u>neither reference</u> teaches a solution that involves <u>receiving a retired instruction</u>, and in response to determining that a previous trace under construction duplicates a trace in a trace cache <u>and that the received instruction corresponds to a branch label</u>, **beginning construction of a new trace.**

The Examiner previously cited Lange's description of the operation of a data cache during fetching from main memory. Appellants asserted that this has absolutely nothing to do with the <u>specific limitations of claim 27</u>. Checking Rotenberg's trace cache for a duplicate trace before collecting a new trace is also contradictory to the Examiner's remarks above regarding the <u>advantages</u> of including duplicate traces. **The Examiner first argued that Rotenberg teaches duplicate traces may exist, and then argued that the combined references teach that the system of Rotenberg should not allow construction of duplicate traces. These arguments cannot coexist if the references are to teach the limitations of claim 27. If no duplicate traces can be constructed, there would be no reason to determine if a trace previously under construction was**

duplicating a trace already in the trace cache. The Examiner's reasons for combining the references are clearly not supported by the cited art, and the references, when combined, do not teach the limitations recited in claim 27.

Appellants note that the Examiner did not provide any additional remarks regarding claim 27 in the Office Action of March 31, 2008. Note also that claim 27 includes limitations similar to those of claim 5 discussed above; therefore the arguments presented above regarding claim 5 apply with equal force to this claim, as well.

For at least the reasons above, the rejection of claim 27 is unsupported by the cited art and removal thereof is respectfully requested.

Claim 35 includes limitations similar to claim 27, and so the arguments presented above apply with equal force to this claim, as well.

<u>Claim 28:</u>

1. The cited art clearly fails to teach or suggest *continuing construction of an incomplete trace already in process in response to determining that the incomplete trace does not duplicate a trace in a trace cache.*

The Examiner previously asserted that Rotenberg teaches this limitation in Section 2.2. Appellants have argued that, as discussed above, the combination of cited references fails to teach or suggest checking the trace cache for a duplicate trace, much less the <u>specific limitations recited in claim 28</u>. In addition, the Examiner has not provided any arguments to support his assertion that this limitation is taught by Rotenberg.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 28. Therefore,

the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

Appellants note that the Examiner did not provide any additional remarks regarding claim 28 in the Office Action of March 31, 2008.

For at least the reasons above, the rejection of claim 28 is unsupported by the cited art, and removal thereof is respectfully requested.

<u>Claim 29:</u>

1. The cited art clearly fails to teach or suggest *searching the trace cache for duplicate entries subsequent to completion of the previous trace under construction or the new trace.*

The Examiner previously submitted that Lange teaches this limitation in column 5, lines 5-10. **Appellants have argued that, as discussed above, Lange teaches nothing about the operation of a trace cache, much less the <u>specific limitations of claim 29</u>.** For example, there is nothing in the combination of cited references that teaches or suggests searching a trace cache for duplicate entries subsequent to completion of the previous trace under construction (i.e., that of claim 27) or the new trace. In addition, the Examiner has not provided any arguments to support his assertion that this limitation is taught by Lange.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 29. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

Appellants note that the Examiner did not provide any additional remarks regarding claim 29 in the Office Action of March 31, 2008.

For at least the reasons above, the rejection of claim 29 is unsupported by the cited art, and removal thereof is respectfully requested.

<u>**Claim 30:**</u>

1. The cited art clearly fails to teach or suggest *creating a new entry in the trace cache in response to no duplicate entry being identified.*

The Examiner previously asserted that Rotenberg teaches this limitation in Section 2.2. Appellants have argued that, as discussed above, the combination of cited references fails to teach or suggest checking the trace cache for a duplicate trace, much less the <u>specific limitations recited in claim 28</u>. In addition, the Examiner has not provided any arguments to support his assertion that this limitation is taught by Rotenberg.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 30. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

Appellants note that the Examiner did not provide any additional remarks regarding claim 30 in the Office Action of March 31, 2008.

For at least the reasons above, the rejection of claim 30 is unsupported by the cited art, and removal thereof is respectfully requested.

<u>**Claim 31:**</u>

1. The cited art clearly fails to teach or suggest *discarding a trace in response to a duplicate entry being identified.*

The Examiner previously submitted that Lange teaches this limitation in column 5, lines 5-10. **Appellants have argued that, as discussed above, Lange teaches nothing about the operation of a trace cache, much less the <u>specific limitations of claim 31</u>.** For example, there is nothing in the combination of cited references that teaches or suggests <u>discarding a trace</u> in response to a <u>duplicate (trace cache) entry</u> being identified. In addition, the Examiner has not provided any arguments to support his assertion that this limitation is taught by Lange.

2. The Examiner has not provided any motivation or other reason to combine the references in teaching the specific limitations of claim 31. Therefore, the Examiner has failed to establish a *prima facie* obviousness of the claimed invention.

Appellants note that the Examiner did not provide any additional remarks regarding claim 31 in the Office Action of March 31, 2008.

For at least the reasons above, the rejection of claim 31 is unsupported by the cited art, and removal thereof is respectfully requested.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-35 was erroneous, and reversal of his decision is respectfully requested.

Since this Appeal Brief is a reinstatement of the previous appeal for which the appeal brief fee has already been paid, no appeal brief fee should be due. The Commissioner is authorized to charge any fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5500-88700/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: September 2, 2008

VIII. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A microprocessor, comprising:

an instruction cache configured to store instructions;

a branch prediction unit;

a trace cache configured to store a plurality of traces of instructions; and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache;

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address;

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address;

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address; and

wherein in response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache.

2. The microprocessor of claim 1, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken.

3. The microprocessor of claim 1, wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction.

4. The microprocessor of claim 1, further comprising a trace generator, wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

5. The microprocessor of claim 4, wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

6. The microprocessor of claim 5, wherein in response to the trace generator identifying a duplicate copy of the trace, the trace generator is configured to discard the trace under construction.

7. The microprocessor of claim 5, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated.

8. The microprocessor of claim 7, wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction.

9. The microprocessor of claim 4, wherein the trace generator is configured to generate traces in response to instructions being retired.

10. The microprocessor of claim 4, wherein the trace generator is configured to generate traces in response to instructions being decoded.

11. The microprocessor of claim 1, wherein each of the plurality of traces comprises partially-decoded instructions.

12. The microprocessor of claim 1, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace.

13. The microprocessor of claim 1, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace.

14. A computer system, comprising:

a system memory; and

a microprocessor coupled to the system memory, comprising:

an instruction cache configured to store instructions;

a branch prediction unit;

a trace cache configured to store a plurality of traces of instructions; and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache;

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address;

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address;

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address; and

wherein in response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache.

15. The computer system of claim 14, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken.

16. The computer system of claim 14, wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction.

17. The computer system of claim 14, further comprising a trace generator, wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

18. The computer system of claim 17, wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

19. The computer system of claim 18, wherein in response to the trace generator identifying a duplicate copy of the trace, the trace generator is configured to discard the trace under construction.

20. The computer system of claim 18, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated.

21. The computer system of claim 20, wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction.

22. The computer system of claim 17, wherein the trace generator is configured to generate traces in response to instructions being retired.

23. The computer system of claim 17, wherein the trace generator is configured to generate traces in response to instructions being decoded.

24. The computer system of claim 14, wherein each of the plurality of traces comprises partially-decoded instructions.

25. The computer system of claim 14, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace.

26. The computer system of claim 14, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace.

27. A method, comprising:

receiving a retired instruction;

determining if a previous trace under construction duplicates a trace in a trace cache and if the received instruction corresponds to a branch label; and

in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label, beginning construction of a new trace.

28. The method of claim 27, further comprising continuing construction of an incomplete trace already in process in response to determining that the incomplete trace does not duplicate a trace in a trace cache.

29. The method of claim 27, further comprising searching the trace cache for duplicate entries subsequent to completion of the previous trace under construction or the new trace.

30. The method of claim 29, further comprising creating a new entry in the trace cache in response to no duplicate entry being identified.

31. The method of claim 29, further comprising discarding a trace in response to a duplicate entry being identified.

32. A method, comprising:

fetching instructions from an instruction cache;

continuing to fetch instructions from the instruction cache without searching a trace cache until a branch target address is generated;

in response to a branch target address being generated, searching a trace cache for an entry corresponding to the branch target address.

33. The method of claim 32, further comprising continuing to fetch instructions from the instruction cache in response to no entry being identified in the trace cache corresponding to the branch target address.

34. The method of claim 32, further comprising fetching one or more traces from the trace cache in response to an entry being identified in the trace cache corresponding to the branch target address.

35. A microprocessor, comprising:

means for receiving a retired operation;

means for determining if a previous trace under construction duplicates a trace in a trace cache and if the received operation is a first operation at a branch label; and

means for starting a new trace in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received operation is a first operation at a branch label.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. <u>RELATED PROCEEDINGS APPENDIX</u>

There are no related proceedings.